Network Services CSC

System Services CSCI

Checkout and Launch Control System (CLCS)

84K00510-010

NOTE: See “**Supporting Document Note**” on following page

PREPARED BY: John Porter, NASA/DP-3 Networks Lead

Steve Davis, NASA/DP-3 Network Services

Supporting Document Note: Acronyms and definitions of many common CLCS terms may be found in the following documents: CLCS Acronyms 84K00240 and CLCS Project Glossary 84K00250.

REVISION HISTORY

REV	DESCRIPTION	DATE
Basic	Redstone post-DP-3 release.	06/19/1997
A	Thor post-DP-3 release.	11/18/1997
A.1	Atlas post-DP2/3 release. Updated document to reflect new requirements and design for the Network APIs—specifically, Reliable Multicast. Revised document to use the CLCS document template.	02/10/1998
B	Promoted per approval by Design Panel. Update to CLCS Format Standards. ljp	5/11/98

LIST OF EFFECTIVE PAGES

Dates of issue of change pages are:

Page No.	A or D*	Issue or Change No.	CR No.	Effective Date**
All	A	Rev. A	N/A	11/18/1997
All	B	Rev. B	N/A	02/10/1998

Table of Contents

1. NETWORK SERVICES CSC

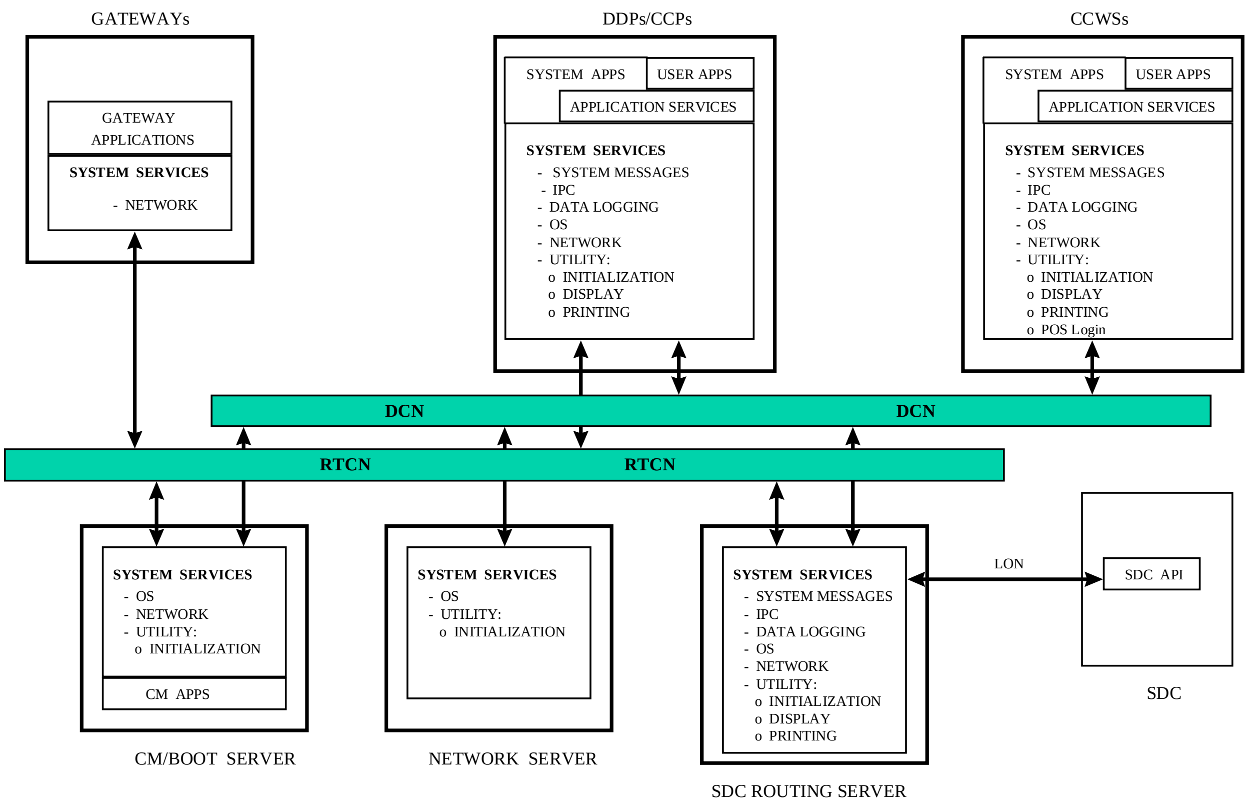
1.1 NETWORK SERVICES INTRODUCTION

1.1.1 Network Services Overview

The Network Services CSC includes the following:

- Basic Communication Service
- Network APIs
- Activity Separation

These services are resident on the RTPS platforms as outlined below.



SYSTEM SERVICES OVERVIEW

Updates to this version of the document are limited to changes in the Network APIs for the Atlas release only. Any impacts to Activity Separation or Basic Communication Services that result from other Atlas threads will be captured in a future version of this document.

1.1.2 Network Services Operational Description

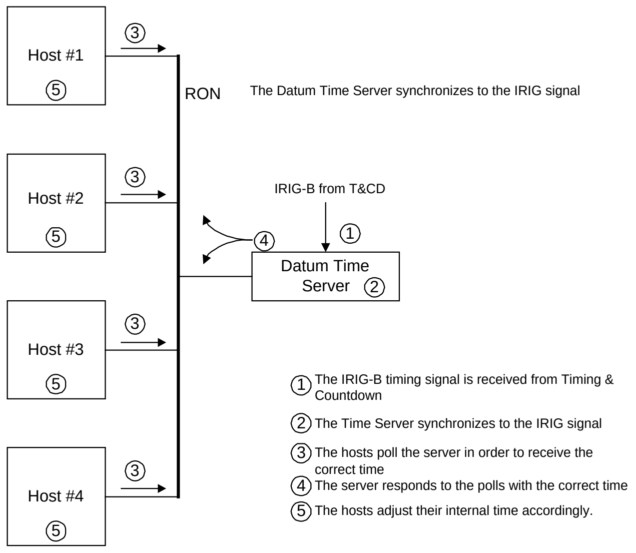
The Network Services provides the capability by which end system applications, services and users communicate and distribute data within CLCS. The three major elements of the Network Services are: Basic Communication Services, Network APIs, and Activities & Activity Separation.

1.1.2.1 Basic Communication Services

The Basic Communications Services, listed below, consist of LAN services and protocols which are supplied by the COTS OS.

- Transmission Control Protocol/ Internet Protocol (TCP/IP): TCP/IP is provided as a COTS product.
- File Transfer Protocol (FTP): The FTP protocol is provided as a COTS product.
- Telephone Network (TELNET): Telnet is provided as a COTS product.
- Network Information Service (NIS): The NIS will be implemented by a NIS server platform that is resident in each Flow Zone. This server will have direct connectivity to each **Development** network (RTCN, DCN) within its associated Flow Zone and any attached Control and Gateway Groups. It will provide password control for users utilizing the resources within the string of equipment.
- Network File System (NFS): The Network File System is provided as a COTS product. It's configuration is controlled by the Operating Systems Group.

- Network Timing Protocol (ntp): The Network Timing Protocol provides a mechanism to synchronize the clocks residing on local hosts with both a common timing source and each other. The common timing source will be a Datum timing box that accepts an IRIG-B timing signal from Timing and Countdown (T&CD) and functions as an ntp server. This Datum box will be present in each Flow Zone and act as the time reference for all local **ntp servers** contained within that Flow Zone as well as any Control Groups that may be attached to that Flow Zone. For Thor, a hierarchical implementation will be considered. Multiple hosts would be selected to synchronize their time directly from the Datum box. The balance of the hosts within a set would then synchronize to those multiple hosts, providing a level of redundancy for ntp. (The choice of hosts to act as ntp servers is an open issue. After the Design Panel discussions, an ERP issue will be opened concerning this subject.)


Host #1
Host #2
Host #3
Host #4
③
⑤
RON
The Datum Time Server synchronizes to the IRIG signal
IRIG-B from T&CD
①
④
Datum Time Server ②
① The IRIG-B timing signal is received from Timing & Countdown
② The Time Server synchronizes to the IRIG signal
③ The hosts poll the server in order to receive the correct time
④ The server responds to the polls with the correct time
⑤ The hosts adjust their internal time accordingly.

Network Timing Protocol Flow
(Redstone only)

- UNIX r-commands (e.g., rlogin): The Unix r-commands are provided as a COTS product.

1.1.2.2 Network APIs

The Network APIs consist of a common set of custom developed library calls which applications can use for communication between computers on RTPS networks.

Application Messaging (AM) consists of a set of APIs for Connection-Oriented Point-To-Point communications. Client Applications will be able to establish and open a connection with applications on a remote server machine. Once the connection is established, data can be sent/received to/from the remote machine.

Connectionless Messaging (CLM) consists of a set of APIs for Connectionless Point-To-Point communications. Local applications will use CLM for sending datagrams Point-To-Point to a specific host machine - no confirmation for transmission or acknowledge of delivery is required from the receiver. For Atlas, RM functionality will be removed from CLM.

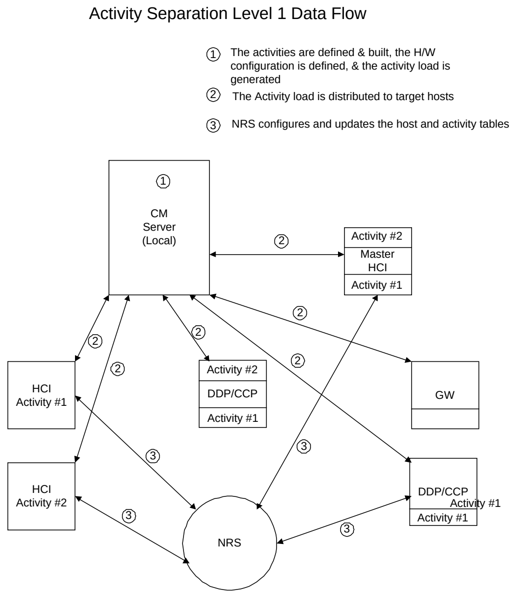
Reliable Multicast (RM) consists of an API and server process that provide reliable Point-To-Multipoint communication. Reliability is achieved both through use of dual, redundant networks (where available), and through retransmission of lost or late messages.

Network Registration Service (NRS). A method for determining the IP assignment of file descriptors, the location of applications and the distribution of this information is required in order for applications to send and receive data on the network. Static registration files or tables will be the mechanism used on the RTCN and for multicast address resolution . The registration files or tables will be maintained through manual procedures. The NRS will be utilized on the DCN for point-to-point communications. NRS allows applications to register their existence by hostname and port number. Other applications can then locate these applications through the NRS APIs. This information is broadcast by each local NRS process onto the network whereby, the NRS process on the other machines update their local NRS information.

1.1.2.3 Activities & Activity Separation

Activity will be a term utilized to classify an operation in the CLCS. A Naming Standard for data streams will be developed which uses a fixed number of generic activity identifiers. Logical Separation can then be made by allowing applications in one Activity to only communicate on their Activity.

An activity will be defined via the OPS CM Activity Manager. When a set of hardware (Gateway Group(s), Control Group(s), Flow Zone(s)) is assigned to support a particular activity, the CM Server will download the activity load to the specific HWCI's supporting that activity utilizing an Activity Manager function. As a part of the load procedure, the host tables for each HWCI will be manipulated (utilizing NRS) in such a manner that only those HWCI's that constitute that particular activity will have visibility to only those HWCI's within their activity. HWCI's that provide services common across activities (ie: NIS servers, etc…) will maintain their capability to see (and be seen by) all other platforms.


Activity Separation Level 1 Data Flow
① The activities are defined & built, the H/W configuration is defined, & the activity load is generated
② The Activity load is distributed to target hosts
③ NRS configures and updates the host and activity tables
①
CM Server (Local)
Activity #2
Master HCI
Activity #1
HCI Activity #1
Activity #2
DDP/CCP
Activity #1
GW
HCI Activity #2
DDP/CCP
Activity #1
Activity #1
NRS
②
③

1.2 NETWORK SERVICES SPECIFICATIONS

1.2.1 Network Services Groundrules

- The Network Services API will be designed with the following RTPS LAN requirements into consideration:

  A) The RTPS LAN will be fault tolerant, such that, a single LAN component will not cause more than one end station failure.

  B) The switching of a LAN component to a backup (due to a failure) will be deterministic and within the Reliable Message time allocation.

  C) Fault isolation and LAN component replacement will be conducted with minimum interference to the on-going operation.

  D) An industry LAN standard protocol for supporting one-to-many communications will be utilized, such that, interoperability among different LAN products can be obtained.

  E) RM will be designed to facilitate troubleshooting of problems that occur during normal operations.

  Note: The exact approach to troubleshooting cannot reasonably be determined until the RM implementation is in place, and a certain amount of experience is gained in its operation.

Whatever approach is ultimately adopted will involve some combination of System Messages, local and remote recording, and System/Subsystem Integrity actions. For now, RM will be designed and implemented so as not to rule out any of these options, but also so that any of them may be disabled with minimal sustaining work if it is deemed necessary in the future.

F) RM will be designed to avoid excessive in-memory copy operations.

- As a part of Redstone and early Thor activities, studies were conducted to examine the current Reliable Messaging implementation. Specifically, the current Library-based implementation was compared to a Process-based implementation as well as comparing the ACK based protocol to multiple protocols (dependent upon the type of traffic being generated) utilizing ACKs or NACKs. The study results were presented to the ERP and a total rewrite of RM was approved. The reason for the approval is that moving from a library to a process and from ACK only to an ACK/NACK based protocol will provide a more efficient use of available bandwidth, more efficient use of CPU resources, and more flexibility in dealing with system messages & error handling. This rewrite of the RM implementation is the primary subject of the current version of this document.

1.2.2 Network Services Functional Requirements

The Network Services Functional Requirements area is composed of the following CSC functions:

1. Network APIs (includes AM/CLM and RM)
2. Network Registration Services API
3. Activity Separation (Data)
4. Activity Separation (Platform)
5. Basic Communications

1.2.2.1 Network APIs

Note: All requirements in this section apply to all RTPS platforms, except as noted in Appendix D of this document.

1.2.2.1.1 Application Messaging and Connectionless Messaging (AM/CLM) Functional Requirements

Note: Requirements marked as *(deleted)* in this section were applicable only to RM functionality, and have been superseded by requirements in section 1.2.2.1.2. See Appendix C for a disposition of previous requirements in this section.

1. Network Services will provide applications with a common API for communicating across the network interfaces.
2. *(deleted)*
3. *(deleted)*
4. The API will be capable of supporting connection oriented point-to-point data, connectionless point-to-point data, and connectionless point-to-multipoint data.
5. The Network Services API will provide a mechanism to associate applications with Service Points (SP’s).
6. The API will support the fragmentation and reassembly of messages that exceed the physical layer

   Maximum Transmission Unit (MTU).
7. *(deleted)*
8. *(deleted)*
9. *(deleted)*
10. *(deleted)*
11. *(deleted)*

12. The API will provide automatic sequential numbering of data packets based on stream or port connections.
13. *(deleted)*
14. *(deleted)*
15. The receiver will report receive errors to the calling application for transfer to System Messages.
16. *(deleted)*
17. Reliable message for connection oriented (point-to-point) communication will be provided as specified by the TCP layer standards.
18. *(deleted)*
19. The API will associate the stream name (defined in the Naming Standard) with a unique multicast address. The stream name assignment to multicast addresses will consist of a static file.
20. The API will allow applications to open multiple streams within a single process.
21. The API will provide a function which relieves applications from having to poll for received data.
22. The API software will be structured and designed such that a single software baseline can be obtained regardless of platform type (i.e. VxWorks or UNIX).
23. *(deleted)*
24. *(deleted)*
25. *(deleted).*

1.2.2.1.2 Reliable Multicast (RM) Functional Requirements

Functional requirements for the RM API are divided into three areas:

1. Requirements applicable to all messages
2. Requirements applicable to non-periodic messages
3. Requirements applicable to periodic messages

1.2.2.1.2.1 General RM Functional Requirements

1. Network Services will provide applications with a Reliable Multicast protocol and API for communicating across the network interfaces.
2. The API shall be capable of reliably sending multicast messages.
3. The API shall provide a function for client processes to open one or more connections, up to a maximum of at least 16 connections per client process and 128 total connections per host.
4. The API shall provide a function for client processes to close connections.
5. The API shall allow a client process to send and receive data to and from multiple computers using a single connection.
6. The API shall allow open connections to be inherited by spawned processes, tasks, or threads.
7. The API shall allow a client process to specify a priority value for each message sent, with a range of at least two different priority levels.
8. The API shall provide a function for client processes to poll for received data.
9. The API shall provide a function which relieves client processes from having to poll for received data.
10. The API shall provide a function for each host to join one or more multicast addresses, up to a maximum of at least 16 addresses.
11. The API shall provide a function for each host to drop multicast addresses that it has previously joined.

12. The API shall provide a function to translate logical destination names into multicast addresses.
13. The API shall provide a function for client processes to register to receive messages on one or more Service Access Points (SAP), up to a maximum of at least 16 SAPs per client process.
14. The API shall allow multiple client processes within a host to register to receive messages on a single SAP.
15. The API shall provide a function to translate well-known logical service names into numeric SAPs.
16. The API shall provide automatic sequential numbering of data messages in order to detect dropped messages.
17. The API shall support the fragmentation and reassembly of messages.
18. The API shall support a maximum total message size of at least 50 kilobytes.
19. The receiver shall discard the whole message if an error is detected or message reassembly is unsuccessful.
20. The API shall report unrecoverable receive errors to the calling client process.
21. The API shall report transmit and receive errors to Subsystem Integrity.
22. The API shall report transmit and receive errors via System Messages.
23. The API shall support activity separation through selective allocations of multicast addresses.
24. The API shall allow a single host to participate in more than one activity.
25. The multicast acknowledgments will be transferred to SDC for recording.
26. The API shall contain the ability to enable/disable the transfer of acknowledgments to SDC for recording, with enable as the default setting.
27. The API shall utilize dual, redundant networks (where available) to increase reliability.
28. The threshold for switching between redundant networks shall be a configurable parameter.

1.2.2.1.2.2 RM Functional Requirements for Non-periodic Messages

1. Network Services shall provide a reliable multicast protocol for non-periodic messages.
2. The API shall provide a function for a client process to optionally designate a single host as the Primary Receiver for messages sent via the non-periodic protocol.
3. The API shall return an error to a sending client process following an unrecoverable failure to transmit messages to a Primary Receiver via the non-periodic protocol.
4. The non-periodic protocol shall support registration of remote receivers for participation in reliable messaging.
5. Messages shall be retransmitted to remote receivers that are registered to receive messages via the non-periodic protocol.
6. The non-periodic protocol shall abort the message send function based on a configurable time-out value set for acknowledgment responses.
7. The number of data message transmission retries for the non-periodic protocol shall be statically configurable per SAP.
8. Re-transmission of the current data message via the non-periodic protocol shall complete or time-out prior to transmission of the next message received from the sending client process.

1.2.2.1.2.3 RM Functional Requirements for Periodic Messages

1. Network Services shall provide a reliable multicast protocol for periodic messages.
2. Messages shall be transmitted via the periodic protocol whether or not any receivers are present.
3. The API shall provide a function for receiving applications to request a retransmission from a sender when it detects that no message has been received within a configurable time period.

4. The number of retransmission requests per periodic message shall be limited by a configurable value.
5. Senders shall hold a configurable number of messages most recently sent via the periodic protocol, in order to handle requests for retransmission by remote hosts.
6. Senders shall retransmit messages requested by a remote receiver, or transmit an error code to the remote receiver if the requested message is no longer available.
7. *(deleted)*

1.2.2.2 Network Registration Service API:

1. The Network Registration Service API will provide the following functions:
   a) A capability that allows applications to add an entry to the registration file.
   b) A locate function that allows applications to locate a registered file entry.
   c) A de-register function that allows applications to remove a registered entry.
2. Registration information will be distributed to all the pertinent platforms assigned to a specific Test Set.
3. All transactions which alter host tables will be logged.
4. Dynamic update of host tables will be provided based on the configured activity on each platform.
5. In point -to-point communications the Network Registration Service will dynamically allocate the port for a given application.
6. Registration of platform name only (without port name) will be permitted.
7. The association of platform and activities will be recorded.
8. The association of platform and activities will be provided through an API.
9. Deactivation of an activity will not interfere with activity resources in use.
10. The following API calls will be used to support registration functions:
    a) register
    b) get port
    c) port register
    d) de-register
    e) search
    f) port search
    g) platform list
    h) activate activity
    i) de-activate activity
    j) get activity
    k) get activity list
11. The Network Registration Service will log all API requests and major network events.
12. The Network Registration Service will provide the capability to register separate point-to-point service IDs by activity.

1.2.2.3 Activity Separation (Data)

1. A naming convention will be provided which maps or is associated to a unique multicast data stream.
2. Receivers will be able to only input the data streams associated with its selected activity.

1.2.2.4 Activity Separation (Platform):

1. Logical separation of platforms will be maintained on the network between different types of activities.

2. A capability will be provided to keep platforms supporting one activity type invisible to platforms supporting another activity.
3. The data separation function will allow certain positions to communicate with devices regardless of the activity selected. Note: These positions are said to be "Global".
4. Separation of point-to-point data will utilize the platform separation.
5. The activity-independent or Global workstation and servers will be visible to all workstations at all times.
6. Uncommitted workstations will not have access to workstations and servers committed to an activity.
7. Each workstation will have access to all workstations and servers which are part of the same activity.

1.2.2.5 Basic Communication Services:

1. The Basic Communication Services will provide the functionality of the following services as defined by the COTS TCP/IP Standards:

SERVICE	Gateway	DDP	CCP	HCI
1.2.2.5.1.1 File Transfer Protocol (ftp)	Yes	Yes	Yes	Yes
1.2.2.5.1.2 Telnet	Yes	Yes	Yes	Yes
1.2.2.5.1.3Unix r-commands	No	Yes	Yes	Yes
1.2.2.5.1.4 Network File System (NFS)	No	Yes	Yes	Yes
1.2.2.5.1.5 Network Timing Protocol (ntp)	No	Yes	Yes	Yes
1.2.2.5.1.6 Network Information Service (NIS)	No	Yes	Yes	Yes

2. A command line interface will be provided for the ftp, Telnet, and Unix r-command services.
3. *The Basic Communication Services function will support the Simple Network Management Protocol (SNMP).*
4. Network Services will provide logical communications such that applications on the same platform can communicate transparently as if they were remote on the network. This is in support of CLCS Application Groupings or Location Transparency onto a single platform

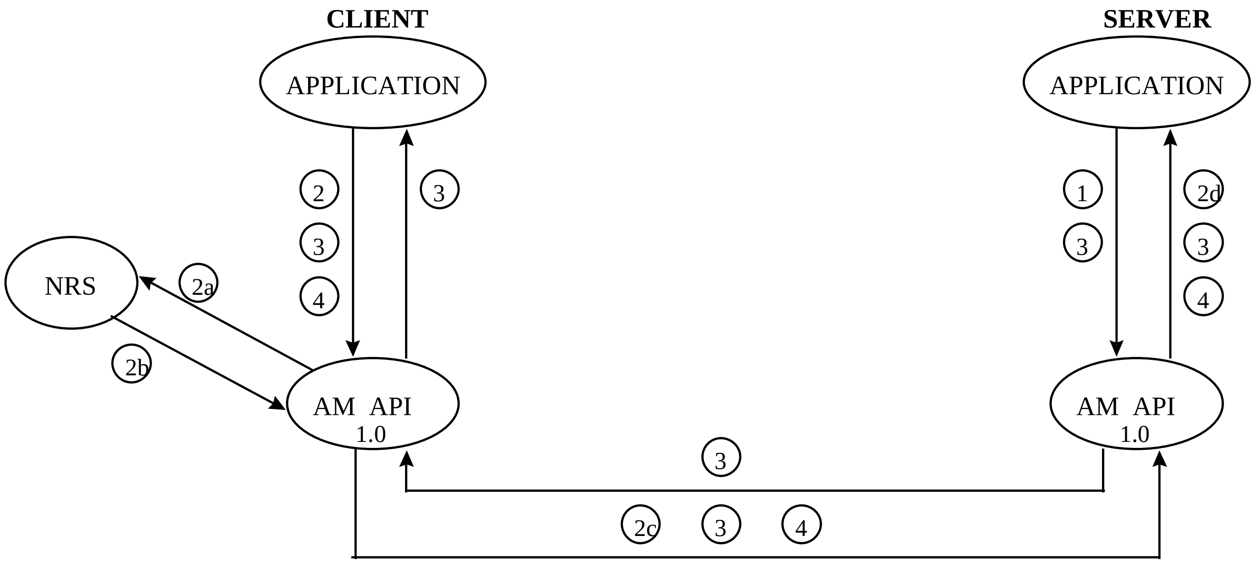
1.2.3 Network Services Performance Requirements

The SLS performance requirements that apply to Network Services are listed in Appendix B. Full performance testing of these requirements will be done as a part of system test. However, as a design goal, RM will be designed to meet the following performance criteria:

1. As a design goal for the RTCN in Atlas, RM shall send a 16-byte, non-periodic message and receive an acknowledgment within one (1) millisecond, in an unloaded system and in the absence of network errors.

1.2.4 Network Services Interfaces Data Flow Diagrams

AM DATA FLOW - LEVEL 1



1 Server Application Opens an AM socket

2 Client Application Opens an AM socket :

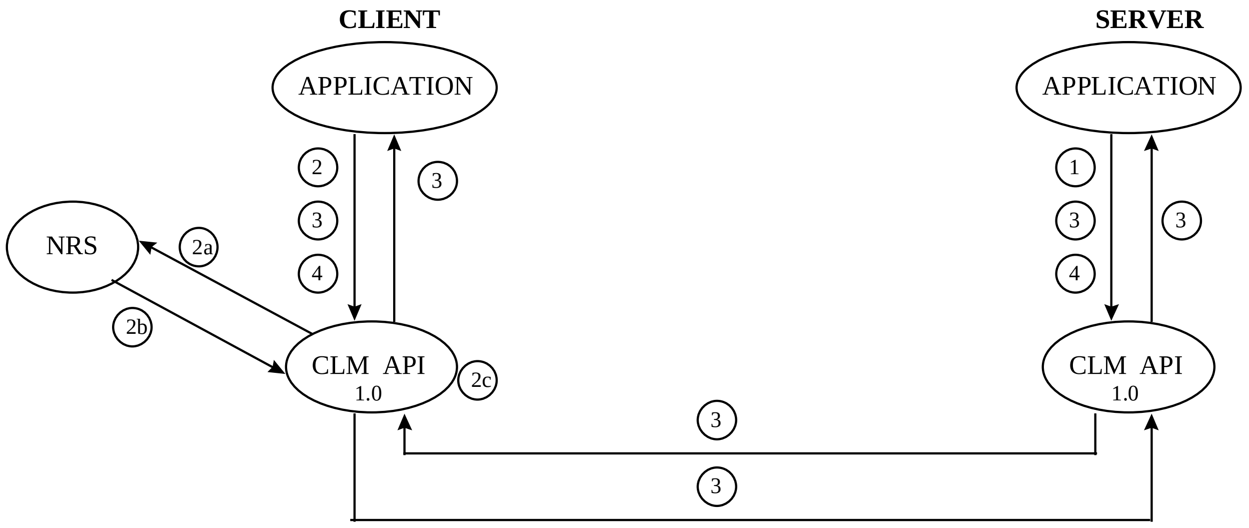
- 2a AM searches NRS for Server location if requested by application
- 2b NRS returns information
- 2c AM connects to server application via TCP
- 2d Server accepts connection socket

3 Client or Server transmits or receives data

4 Client closes connection and AM notifies the server process

CLM DATA FLOW - LEVEL 1

(UNRELIABLE POINT-TO-POINT & BROADCAST)



1 Server Application Opens a CLM P-P or Broadcast socket

2 Client Application Opens a CLM P-P or Broadcast socket :

2a CLM searches NRS for P-P Server location if requested by application

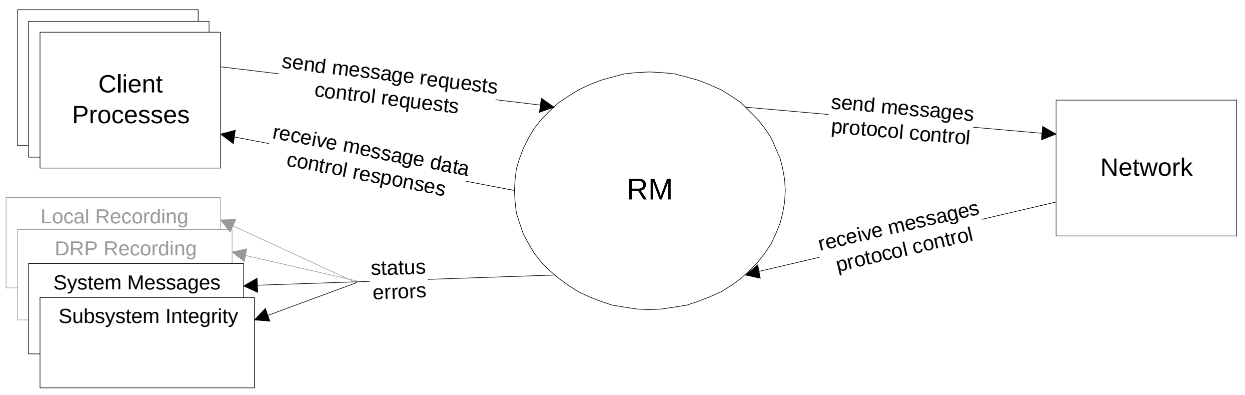
2b NRS returns P-P information

2c Client API sets up destination address

3 Client or Server transmits or receives data

4 Each appliation closes the CLM socket

Reliable Multicast Top-Level Data Flow Diagram



Notes:

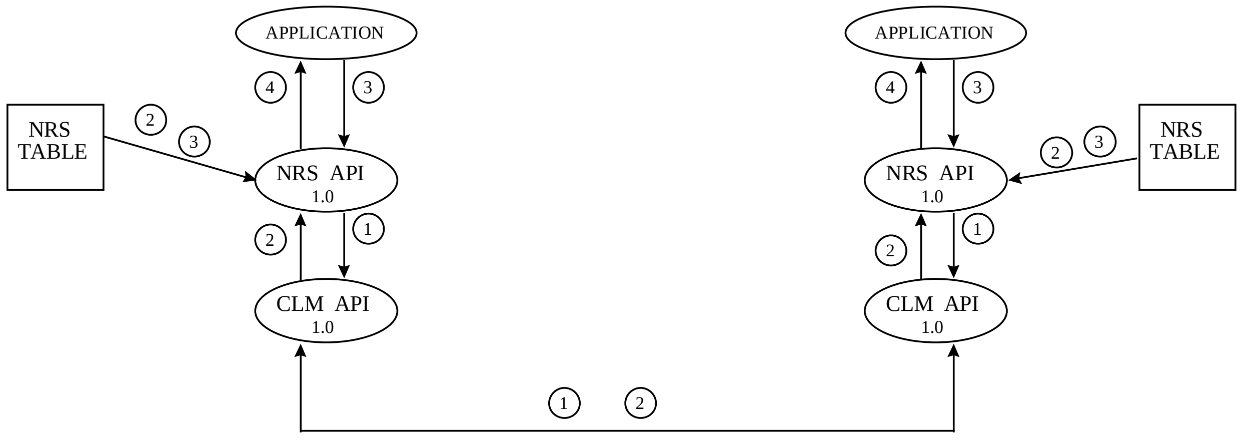
a) Client processes access RM through a set of API calls (see sect. 1.3.2.7). Status and errors that are of concern to client processes are reported through this API.

b) RM reports status and errors via the System Messages (SMS) and Subsystem Integrity (SSI) APIs.

c) Interfaces for Local Recording and DRP (SDC) Recording are not defined at this time. RM is designed so that support for these interfaces may be easily added in the future.

d) RM uses the standard Sockets interface to CLCS networks.

NRS DATA FLOW - LEVEL 1



(1) Each NRS Server broadcasts its registered Service IDs + Activity data via CLM

(2) Each NRS Server receives all NRS broadcasts and updates its local memory NRS Table

(3) Applications call NRS APIs to add, delete, or search the NRS data

(4) NRS returns status or data

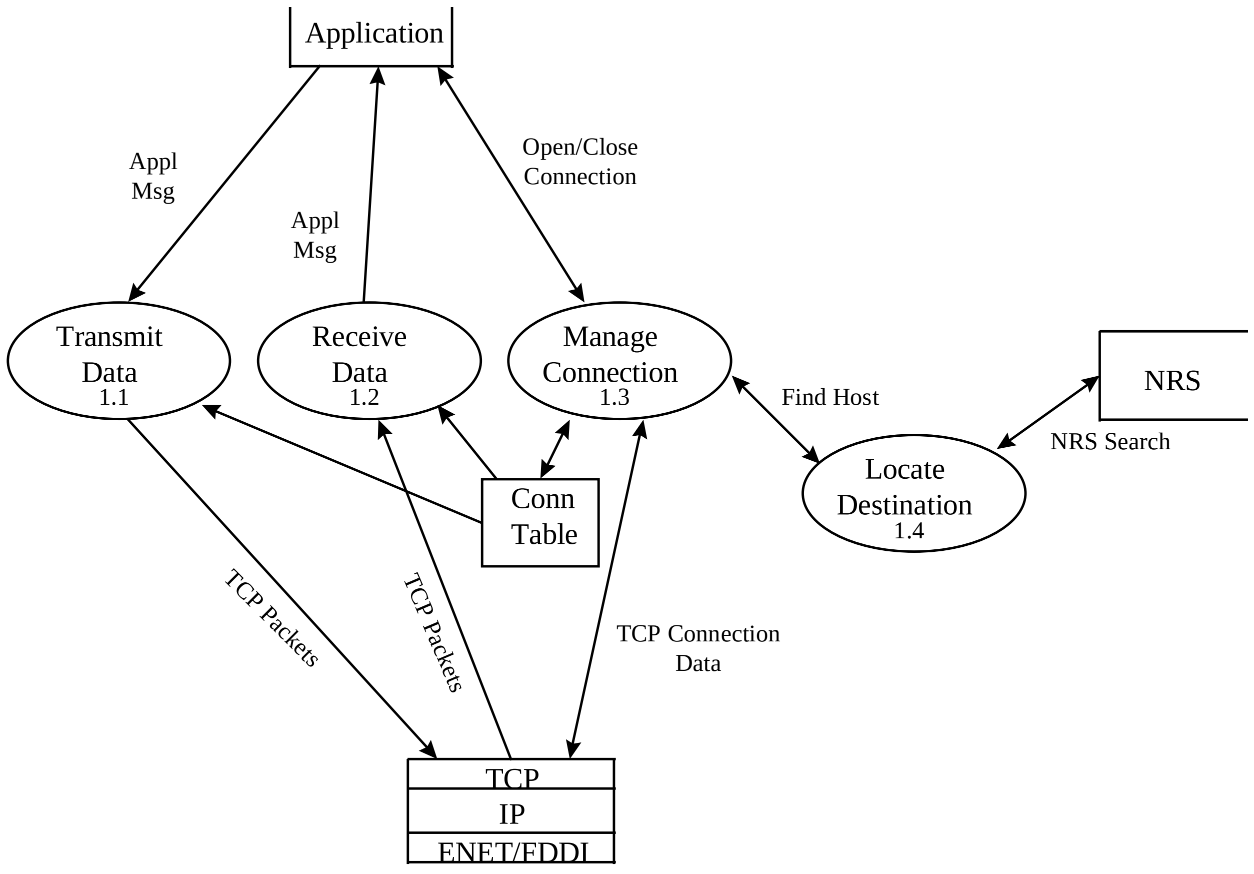
1.3 NETWORK SERVICES DESIGN SPECIFICATION

1.3.1 Network Services Detailed Data Flow

1.3.1.1 AM/CLM Detailed Data Flow

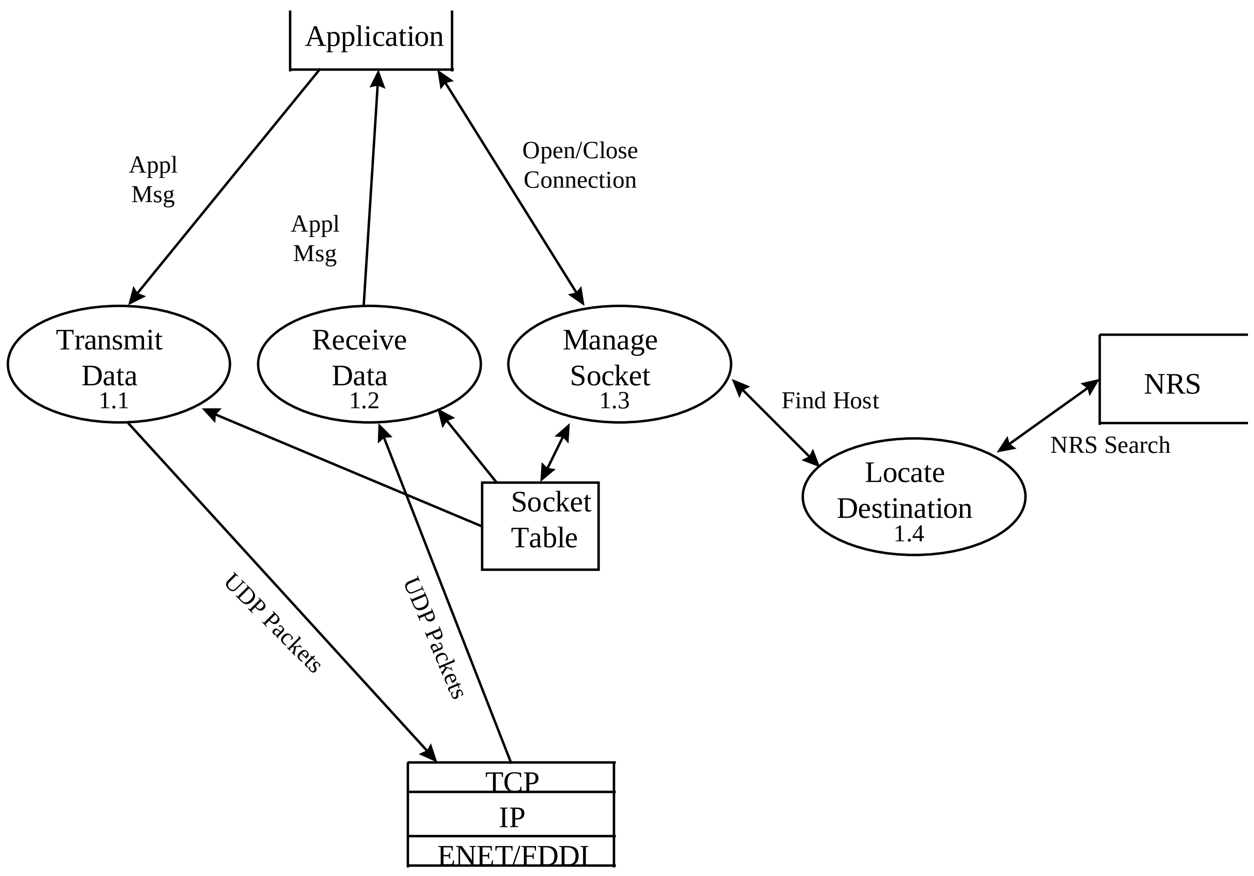
APPLICATION MESSAGING

DATA FLOW DIAGRAM - LEVEL 2



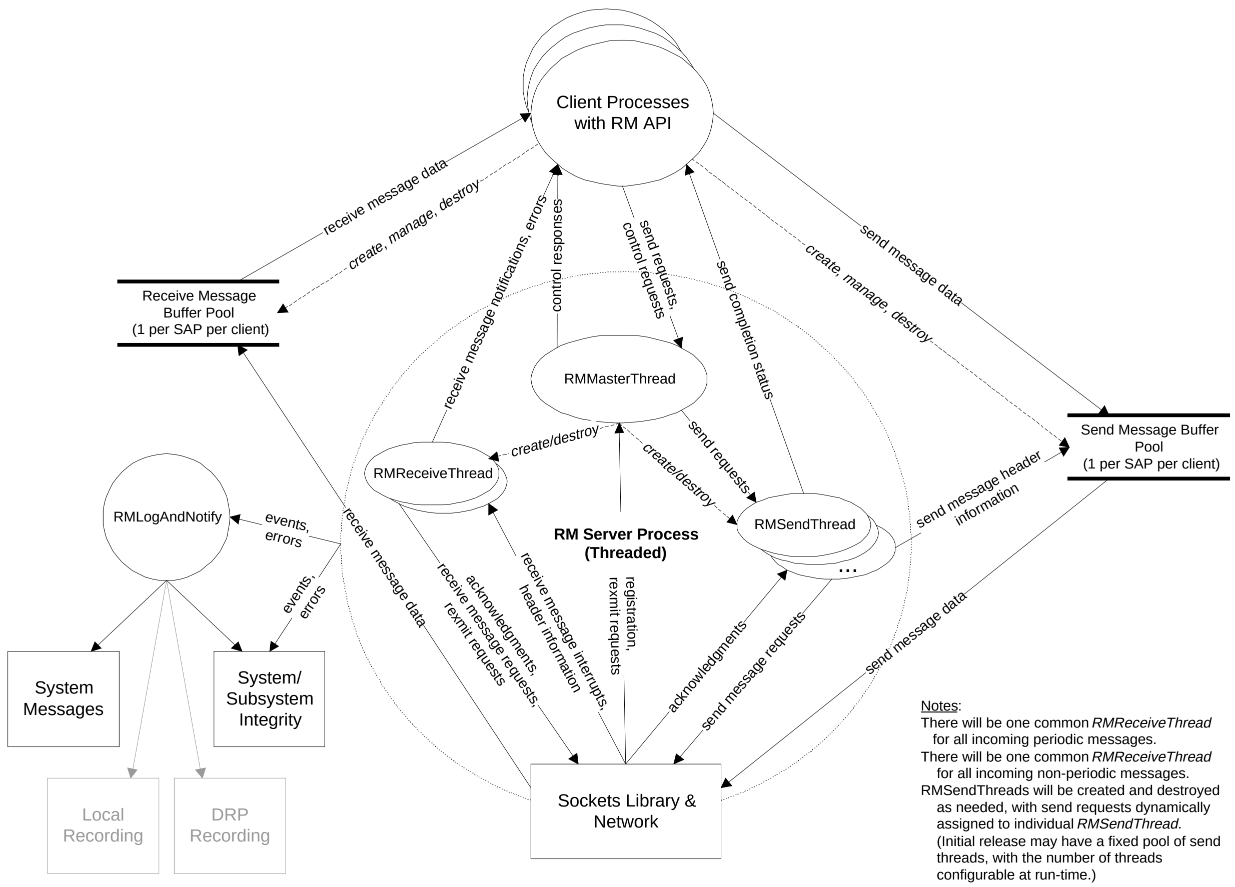
CONNECTIONLESS MESSAGING (Broadcast, Point-to-Point)

DATA FLOW DIAGRAM - LEVEL 2



1.3.1.2 Reliable Multicast (RM) Detailed Data Flow

The RM process architecture and data flow is captured in the following diagram:


Client Processes with RM API
receive message data
create, manage, destroy
receive message notifications, errors
control responses
send requests, control requests
send completion status
send message data
create, manage, destroy
Receive Message Buffer Pool (1 per SAP per client)
RMMasterThread
create/destroy
create/destroy
send requests
Send Message Buffer Pool (1 per SAP per client)
send message header information
RMReceiveThread
RMSendThread
...
RMLogAndNotify
events, errors
events, errors
RM Server Process (Threaded)
receive message data
acknowledgments, receive message requests, rexmit requests
receive message interrupts, header information
registration, rexmit requests
acknowledgments
send message requests
send message data
System Messages
System/ Subsystem Integrity
Local Recording
DRP Recording
Sockets Library & Network

Notes:
There will be one common *RMReceiveThread* for all incoming periodic messages.
There will be one common *RMReceiveThread* for all incoming non-periodic messages.
RMSendThreads will be created and destroyed as needed, with send requests dynamically assigned to individual *RMSendThread*. (Initial release may have a fixed pool of send threads, with the number of threads configurable at run-time.)

1.3.1.2.1 RM Process Architecture Overview

The Atlas release of RM is designed with a client-server architecture within each RTPS computer. Client processes (i.e., system and user applications) communicate over RTPS networks (RTCN and DCN) through the RM Server on the local computer. Communication between computers is handled by the RM Server on each computer in a peer-to-peer manner.

For DDP, CCP, and CCWS platforms, the RM Server is implemented as a threaded process, indicated by the large dotted-line circle in the above DFD. For Gateway platforms, the RM Server is implemented as a group of interrelated tasks, the functions of which map closely to the threads in the DDP/CCP/CCWS implementation.

Clients interact with the RM Server through the RM API. This API provides a library of ‘C’ functions to open and close connections to the RM server, send and receive messages, manage send and receive buffers, and add and drop message streams. The API also provides functions for the Operations Configuration Manager to configure the logical identity of the RM Server for the local computer. The RM API is summarized in section 1.3.2.7 of this document.

A key feature of the Atlas release of RM is the avoidance of excessive in-memory copy operations during the normal processing of messages. To accomplish this, the RM API requires clients to build messages to be sent in shared memory buffers, then refer the RM Server to these buffers for transmittal. Similarly, the RM server places received messages in shared memory buffers, which are then referred to the client. The exception to this copy avoidance scheme is when more than one client process wishes to receive messages on the same SAP within the same computer. In this case, the RM server produces copies of received messages in each client's shared memory buffer.

Interfaces to external services such as local and remote recording, System Messages, and Subsystem Integrity is accomplished through the RM Log-and-Notify process. This process provides notification of events and errors encountered during normal execution of RM, both for near-real-time operator notification and for off-line troubleshooting. (Note: The RM Server may interface directly to Subsystem Integrity rather than using the Log-and-Notify process, for performance reasons.)

1.3.1.2.2 RM Concept of Operations

Key features of RM are:

- Use of dual, redundant networks (where available) to increase reliability.
- Assignment of logical host identifiers to each RTPS computer.
- Use of separate protocols to support periodic and non-periodic messages.

The design of these features is described in this section.

Redundant Network Utilization

In CLCS sets that are implemented with redundant networks for the RTCN, RM can operate in one of two modes:

1. Messages and acknowledgments are transmitted over the primary network unless time-outs or retransmission requests are encountered. Retransmission of messages and retransmission requests are sent over both networks. When the RM Server on a computer detects that the primary network is not functioning properly, it switches the roles of the primary and backup networks. (The concept of a "primary" and "backup" network, and of switching their roles, is local to each RTPS computer.)
2. Sending computers transmit all messages over both networks, and receivers accept the first copy of a message received over either network. Acknowledgments and retransmission requests are also sent over both networks. In this mode, there is no primary or backup network.

To facilitate maintenance of network hardware, the RM API will allow configuration of the RM Server to disallow automatic network switch-over, and to use only one network. In both modes, the RM Server will monitor both the primary and backup networks for incoming messages.

The purpose of this dual-mode implementation is to allow maximum flexibility in tuning CLCS as more operational experience is gained.

Logical Host Identifiers

Each computer within a test set is assigned a logical host identifier, which when combined with the test set identifier, maps to a unique multicast address monitored by that computer as its primary address. Messages intended for a particular computer are sent to this address. This accomplishes two objectives:

1. Other computers within the test set may reliably monitor messages sent to another computer, but may not impact the client application sending such messages. (Examples: DRP; standby member of an active/standby pair)
2. Sending hosts will receive positive confirmation that a message sent to a particular computer was received by that computer, regardless of any other computers that did or did not receive it.

RM will also allow the transmission of messages to addresses that are not bound to any one computer. (Examples: DDP-generated change data; DCN System Messages)

Non-Periodic Message Protocol

For non-periodic messages, each message sent generates an acknowledgment from all computers registered to receive messages from that sender/SAP combination. A receiver may become registered by one of two means:

1. The receiver is the primary receiver to which the message is addressed, based on its logical host identifier. In this case, the receiver is inherently registered.

2. The receiver has informed the sender that it wishes to reliably receive messages addressed to another receiver's logical host identifier. In this case, the sender maintains a list of registered receivers, and whether each has acknowledged the current message.

After the RM Server sends a non-periodic message, it waits for acknowledgments from all registered receivers for a configurable time-out period. If all expected acknowledgments are not received by the end of this period, the message is retransmitted and a new time-out period begins. Success status is returned to the sending client application immediately when an acknowledgment is received from the primary receiver of the message. However, the RM Server continues the retransmit cycle until all expected acknowledgments are received, or until a configurable number of retransmissions have occurred. (Performance note: the worst case blocking time for sending a message is the time-out value multiplied by the retransmit count, plus processing overhead. These parameters must be chosen wisely to ensure adequate performance.)

Periodic Message Protocol

For periodic messages, the RM Server returns success to the sending client process immediately following successful transmission onto the network. However, the RM Server buffers a configurable number of previously sent messages in order to service retransmission requests.

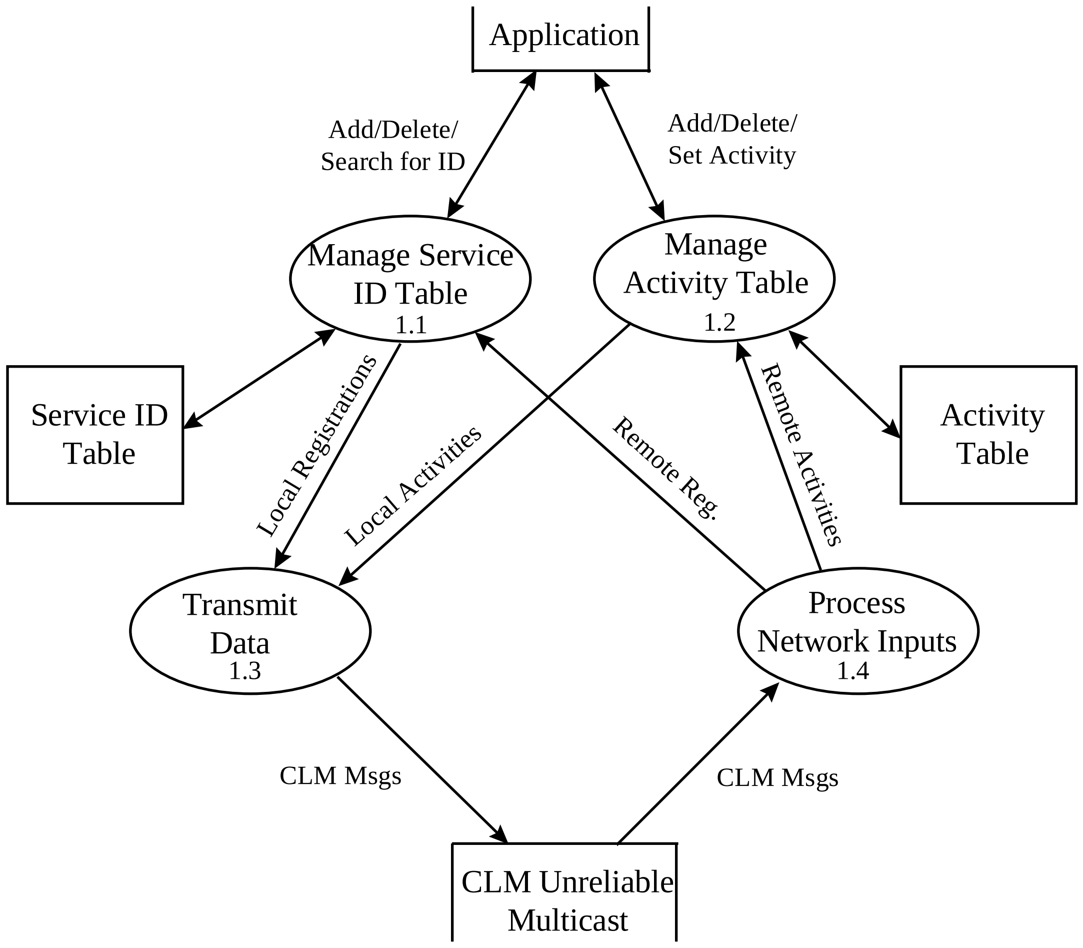
Sending client applications are responsible for queuing outgoing messages to the RM Server at the periodic rate for that stream. Similarly, receiving client applications are responsible for detecting when an expected message has not been received on time, and requesting a retransmission through the RM Server. The RM API provides a function for this purpose.

The RM Server on receiving platforms detects breaks in the sequence of received messages. In this case the RM Server automatically requests a retransmission from the sending computer.

Upon detecting a missed message or upon a retransmission request from a receiving client application, the RM Server on the receiving computer sends a retransmission request to the sending computer. The RM Server on the sender either retransmits the message if it is still available, or transmits a "message no longer available" code to the requesting receiver. In the latter case, an error status is returned to the receiving client application.

1.3.1.3 Network Services (NRS) Detailed Data Flow

NETWORK REGISTRATION SERVICE
DATA FLOW DIAGRAM - LEVEL 2



1.3.2 Network Services External Interfaces

1.3.2.1 Network Services Message Formats

1.3.2.1.1 AM/CLM Message Formats

Since this CSC is strictly a set of library calls provided for the use of calling applications, no System Messages will be generated by the Network Services CSC. Rather, error codes will be generated and passed to the calling applications. The rationale for this implementation is that the overhead required to set up a System Message connection and to generate packets for transmission is beyond the scope of a library call.

```
Enoerr = 0,        /*  0 Error 0 */
Einvalid,          /*  1 invalid parameter */
Eopenclntsoc,      /*  2 open client socket failed */
Econnect,          /*  3 connect(2) to server failed */
Econntimeout,      /*  4 connect timed out */
Eopenservsoc,      /*  5 open server socket failed */
Elisten,           /*  6 listen failed */
EUDPrdwldblk,      /*  7 UDP read of next packet in this message would block */
Esendto,           /*  8 UDP sendto failed */
Eservbind,         /*  9 bind(2) of the socket failed */
Eaccept,           /* 10 accept(2) failed */
Eread,             /* 11 read(2) failed */
Erdbufsmall,       /* 12 incoming message is larger than the receive buffer */
Ewrite,            /* 13 write(2) failed */
Ewritev,           /* 14 writev(2) failed */
Econnwldblk,       /* 15 connect call would block */
Erecvfrom,         /* 16 UDP recvfrom failed */
Eservnotfound,     /* 17 service not found */
Enodenotfound,     /* 18 node not found */
EMCservice,        /* 19 Multicast service not found in IP MC table */
EMCtable,          /* 20 Could not open IP Multicast table */
Eiddselect,        /* 21 select(2) failed in idd_select */
ETCPrdwldblk,      /* 22 TCP read would block */
ETCPwrtwldblk,     /* 23 TCP write would block */
Emalloc,           /* 24 malloc of send buffer space failed */
Epipe,             /* 25 write on a socket with no one to read it */
Eeatmessage,       /* 26 receive buffer too small and attempt to discard extra bytes failed */
Esetrcvbuf,        /* 27 attempt to increase socket receive buffer size failed */
Esetsndbuf,        /* 28 attempt to increase socket send buffer size failed */
Enewsock,          /* 29 attempt to open new socket for UDP segmenting failed */
Esendnewsoc,       /* 30 attempt to send new socket address for segmenting failed */
Ewrongpacket,      /* 31 packet received was not part of message currently being assembled */
Erecvsocwldblk,    /* 32 segmenting UDP message and receive of new server socket timed out */
Erecvnewsoc,       /* 33 attempt to receive new socket address for segmenting failed */
Eclntbind,         /* 34 bind(2) of UNIX client name failed */
EUDPselect,        /* 35 using select(2) to delay between UDP segments failed */
Eaccptwldblk,      /* 36 accept(2) would block */
EUDPsendwldblk,    /* 37 UDP sendto(2) would block */
EUDPrecvwldblk,    /* 38 UDP recvfrom(2) would block */
Eselectintr,       /* 39 select(2) was interrupted by a signal */
ENRSsearch,        /* 40 NRS search for a host for this service failed */
EMCopen,           /* 41 open of IP multicast failed */
```

```
EMCioctl,        /* 42 ioctl(2) for LAN device failed */
Eclose,          /* 43 close(2) of socket descriptor failed */
Eunlink,         /* 44 unlink(2) of Unix socket name failed */
Einprogress,     /* 45 The connection is requested on a socket with FNDELAY set */
Elostbytes,      /* 46 client disconnected before sending all expected bytes */
EBCsocket,       /* 47 socket(2) for broadcast failed */
EBCenable,       /* 48 setsockopt(2) to enable broadcast failed */
Etmpsock,        /* 49 socket(2) for temp socket failed */
Esockrange,      /* 50 socket descriptor is greater than AM/CLM MAX_OPEN_SOCKETS */
EInvalidIPMC,    /* 51 Conversion of IP multicast address to network address failed */
ENoIPMCTable,    /* 52 IP MUlticast table was not found */
EReuseAddr,      /* 53 setsockopt(2) for REUSEADDR failed */
EIntfNotFound,   /* 54 LAN interface not found */
EReUsePort,      /* 55 setsockopt(2) to enable REUSEPORT failed */
ESetMCaddr,      /* 56 setsockopt(2) to enable IP MC address failed */
EIP_MCgroup,     /* 57 setsockopt(2) to join or leave a multicast group failed */
```

1.3.2.1.2 Reliable Multicast (RM) Message Formats

API function return values will be documented in detail in the Network Services API Manual.

RM provides the capability to generate system messages in response to events and errors encountered during operation. These messages include:

- RM fatal error encountered: condition code.
- Retry limit reached: source, destination, SAP, sequence number.
- Recoverable time-outs exceed threshold: source, destination, SAP.
- Data loss: source, destination, SAP, last good sequence number, current sequence number

1.3.2.2 Network Services Display Formats

This section is not applicable to the Network Services CSC

1.3.2.3 Network Services Input Formats

This section is not applicable to the Network Services CSC.

1.3.2.4 Recorded Data

This section is not applicable to the Network Services CSC.

1.3.2.5 Network Services Printer Formats

This section is not applicable to the Network Services CSC.

1.3.2.6 Interprocess Communications (C-to-C Communications?)

All network services IPCs are hidden within the network services API calls, and within API calls of other CSCIs with which RM interfaces. Refer to the data flow diagrams in Section 1.3.1 of this document for details. (Note: RM IPCs do not use the CLCS C-to-C format. RM is the transport service through which C-to-Cs are sent by higher-level CSCIs.)

1.3.2.7 Network Services External Interface Calls (e.g., API Calling Formats)

The Network Services API is fully defined in the **Network Services Interface Definition Document (84K00354)**. Provided below is a complete list of the available calls with their parameters. For a full definition, refer to the IDD.

AM/CLM Calls:

- am_open(node, service, type)
- am_connect(sd)

- am_accept(sd)
- am_close(sd)
- am_recv(sd, bufptr, bufsize, log)
- am_send(sd, bufptr, nbytes, log)
- am_get_client(sd)
- am_set_timeout(t)
- clm_open(node, service, type)
- clm_close(sd)
- clm_recv(sd, bufptr, bufsize, log)
- clm_send(sd, msgptr, msgsize, log)
- clm_get_addr(sd)
- clm_set_addr(sd, sp)
- idd_set_bufsize(sd, type, rsize, ssize)
- idd_select(sd, sel, timeout)

NRS Calls:

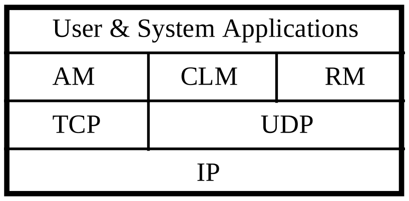
- nrs_register(service_id)
- nrs_get_port(service_id, port_id)
- nrs_port_reg(service_id, port_id)
- nrs_deregister(service_id)
- nrs_server_deregister(service_id)
- nrs_search(service_id, host_ids)
- nrs_port_srch(service_id, host_ids, port_ids)
- nrs_node_list(node_ids, activity_ids)
- nrs_activate_act(activity)
- nrs_deactivate_act(activity_id)
- nrs_get_act(index, activity)
- nrs_get_act_table(activity)
- nrs_ws_config_act(activity_id)
- NRS_aix_gethostname() **(obsolete)**
- nrs_remote_port_reg(service_id, host_id, act, port_id)
- nrs_remote_cmd(service_id, host_id, cmd_type, act)
- nrs_service_name_srch(service_name, host_ids, serv_ids, port_ids)
- nrs_act_type_node_srch(option, host_ids, act_types, activity_ids)

RM Calls:

- RMOpen (ServerID)
- RMClose (ConnectionID)
- RMDisconnect (ConnectionID)
- RMAddGroup (ConnectionID, GroupID, Flags)
- RMDropGroup (ConnectionID, GroupID)
- RMAddActivity (ConnectionID, ActivityID, Flags)
- RMJoinStream (ConnectionID, StreamID)
- RMDropStream (ConnectionID, StreamID)
- RMGetServiceByName (ServiceName)
- RMAddService (ConnectionID, SAP, Flags)
- RMDropService (ConnectionID, SAP, Flags)
- RMGetBuffer (ConnectionID, SAP)
- RMFreeBuffer (ConnectionID, SAP, BufferID)
- RMSend (ConnectionID, LogicalDestination, SAP, BufferID, Priority)
- RMReceive (ConnectionID)
- RMRetransmit (ConnectionID, LogicalSource, SAP, LastMessageReceived)
- RMPoll (ConnectionID)

- RMNotify (ConnectionID, SignalID)
- RMGetAttributes (ConnectionID)
- RMSetAttributes (ConnectionID, AttributeFlags)

The relationship to the AM, CLM, and RM API calls to client applications is depicted in the following simplified software hierarchy diagram:



1.3.2.8 Network Services Table Formats

1.3.2.8.1 AM/CLM Table Formats

The RM Static Address Table from Thor and previous releases will be retained solely for continued support of the CLM Unreliable Multicast (IP_MC_SndRcv) option. The format for entries in this table is simplified to the following:

<stream_name> <data_ip_address> <data_udp_port >

where:
stream_name is the name of the data stream (this format is TBD),
data_ip_address is the address of the data stream (broadcast or multicast),
data_udp_port is the UDP port number associated with the data stream,

This static file must be present on all hosts using the CLM unreliable multicast protocol.

1.3.2.8.2 RM Table Formats

RM will take input from an externally generated table associating each Service Access Point (SAP) name with a numeric SAP ID and parameter values unique to that SAP. This table will contain the following information for each SAP:

SAP name
Numeric SAP ID
Service Type (periodic or non-periodic)
Maximum message size
Maximum number of messages outstanding
Time-out (for non-periodic ACKs and periodic messages)
Retransmission limit

The SAP table is generated by hand. It is expected to remain relatively small and static, compared to the Static Address Table from previous releases.

1.3.3 Network Services Test Plan

Testing of the Network Services CSC for this release is limited to testing of the Network APIs.

1.3.3.1 Test Environment

Test cases not requiring redundant networks will be conducted in the HMF set. Test cases requiring redundant networks will be conducted in the SDE-2 set. All processor platforms and network switches that will be required are available as part of these sets. Each system involved shall be loaded with a baseline Operating System, the network API's, and any test scripts to be executed. In addition, the network elements (i.e. switches) will be loaded with a baseline configuration prior to testing.

1.3.3.2 Test Tools

In addition to the working environment of equipment where network services reside, up to two LAN analyzers may also be required for some test procedures. These analyzers will need the ability to capture individual network frames or cells as well as to re-assemble these units into packets from higher level COTS protocols.

Custom test software will be written to test the reliable multicast capability. This software will simply allow a number of messages to be transmitted and received with the following parameters:

- Number of messages
- Size of messages
- Delay between successive messages

Parameters such as time-out and retransmission count will be established using standard Network Services facilities.

1.3.3.3 Test Cases

The testing will be broken down into the following phases:

- Functionality Testing - test the API's adherence to the functional requirements, per Design Panel 2.
- Error Reporting to Application Testing - test the API's reporting of error conditions to calling applications.
- Error and Event Reporting to O&M Testing - test the API's reporting of events and errors to O&M through appropriate methods.
- Limit Testing - test the API's performance with data of minimum and maximum values prescribed in the SLS.
- Performance Testing - test the API's adherence to the timing performance prescribed by the SLS.
- Stress Testing - determine if there is a point at which the Network Services API fails to function properly. This testing will be performed by the System Stress Test Pathfinder.

APPENDIX A

Statement of Work

APPENDIX A: STATEMENT OF WORK

The following is a list of Statement of Work items from the Atlas Reliable Messages Thread Design Panel 1. This information is provided for reference only.

- The stream naming and address mapping convention of previous releases will be abandoned. In Atlas, messages will be sent to Logical Destinations, which in most cases will map to a logical host within the test set. Hosts other than the host assigned to the Logical Destination will be able to monitor messages sent to that destination and participate in reliable messaging. The capability will be provided to create Logical Destinations with no host association, when appropriate (e.g.: Data Distribution, System Messages). Within a given test set or activity, Logical Destinations will map one-to-one with multicast addresses.
- Activity Separation will be supported based on a numerical activity ID. Each activity ID will map to a private range of multicast addresses for that activity.
- Multiple activities will be supported within Unix-based RTPS subsystems. There is no requirement to support multiple activities on Gateway subsystems.
- A server process will be implemented to handle the real-time exchange of data and acknowledgments on behalf of client applications.
- Both periodic and non-periodic message streams will be supported with efficient protocols for each stream type.
- As a design goal, the number of in-memory copy operations during normal message processing will be minimized.
- Updates of errors and events will be provided for use by Subsystem Integrity.
- Adequate information will be provided to O&M for notification and troubleshooting of RM events and errors.
- All applicable SLS performance requirements will be supported. Performance budgets for RM based on these SLS requirements will be generated in coordination with SE&I.

APPENDIX B

SLS Requirements

APPENDIX B: APPLICABLE SLS REQUIREMENTS

The following is a list of SLS requirements applicable to the Network Services CSC, as approved at the Atlas Reliable Messages Thread Design Panel 1. These requirements are provided here for reference only.

(2.2.1.1.2) The RTPS shall be fault tolerant. Specifically, the system shall provide the capability to recover from subsystem failures in the following areas:

4. Real Time Critical Network and the Display and Control Network

(2.2.1.1.3) The CLCS shall be designed to have a high level of data integrity. Specifically, the system shall provide the following:

1. No loss of command data within the CLCS
2. No loss of measurement data within the CLCS
4. No data which has been corrupted within the CLCS

(2.2.1.1.4) The RTPS shall provide fault tolerance in the Command and Control HCI positions.

(2.2.1.1.5) The loss of any RTPS Real Time Network component shall not cause switchover of more than one standby subsystem.

(2.2.2.1.1) The system shall support 25,000 End-Item Function Designator changes per second continuously. This is the “system maximum data bandwidth”.

(2.2.2.1.2) The system shall support a peak of 50,000 End-Item Function Designator changes in a given second without losing any data.

(2.2.2.1.3) The system shall support 1,000 End-Item Function Designator changes during a 10 millisecond period.

(2.2.2.1.5) The system shall execute a GSE reactive sequence, which simply tests 10 FDs, changes the constraints on 10 FDs and responds to the notification with a command, in less than 100 milliseconds from the time the measurement (1000 samples per second) changes at the HIM input until the command is issued at the HIM output while the system is at the “system maximum data bandwidth” and the subsystem executing the sequence has 20 test applications executing.

(2.2.2.1.6) The system shall execute a GSE control loop test application, which simply responds to the notification with a command, in less than 100 milliseconds from the time the measurement (1000 samples per second) changes at the HIM input until the command is issued at the HIM output in an unloaded system with no other test applications executing in the subsystem executing the control loop.

(2.2.2.1.8) GSE command/response latency of a priority command, or of a non-priority command in an unloaded system, shall be less than 20 milliseconds from the time a test/control application issues the command until the response is received by the test application.

(2.2.2.1.9) GSE command/response latency of a non-priority command in a system supporting the “system maximum data bandwidth” and with 20 test applications executing in the same subsystem shall be less than 100 milliseconds from the time the test/control application issues the command until the response is received by the test application.

(2.2.2.1.14) The system shall support executing a manual command in less than one second from a human execution to RTPS interface output.

(2.2.2.1.21) The system shall support (while executing at 40 percent of the system maximum data bandwidth) 25 Constraint Management requests from each of 4 Command and Control Processor Subsystems (100 total), 20 constraint event notifications (5 per CCP), and 4 commands (1 per CCP) with each CCP executing 20 User Test Applications which are executing 500 Application Service calls per second.

(2.2.3.1.5) The RTPS shall provide changed measurement data to system and user applications at the System Synchronous Rate.

(2.2.3.1.6) The RTPS shall provide changed measurement data to display applications at the Display Synchronous Rate.

APPENDIX C

Disposition of Previous Network Services Requirements

APPENDIX C: DISPOSITION OF PREVIOUS NETWORK SERVICES REQUIREMENTS

Thor Requirement	Disposition	Comments/Rationale
1. Network Services will provide applications with a common API for communicating across the network interfaces.	AM/CLM, RM	Retained for AM/CLM. Rewritten as 1.2.2.1.2.1-1 for RM.
2. The Network Services API will provide independent transmit and receive functionality.	DELETED	Not applicable to AM/CLM. Does not map to RM's new address assignment scheme.
3. The API will be capable of reliably sending multicast and broadcast data streams.	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.1-2 for RM.
4. The API will be capable of supporting connection oriented point-to-point data, connectionless point-to-point data, and connectionless point-to-multipoint data.	AM/CLM	Applicable only to AM/CLM. RM does not send point-to-point data.
5. The Network Services API will provide a mechanism to associate applications with Service Points (SP's).	AM/CLM, RM	Retained for AM/CLM. Rewritten as 1.2.2.1.2.1-13 for RM.
6. The API will support the fragmentation and reassembly of messages that exceed the physical layer Maximum Transmission Unit (MTU).	AM/CLM, RM	Retained for AM/CLM. Rewritten as 1.2.2.1.2.1-17 for RM.
7. Data messages will be re-transmitted to end stations registered for reliable message communication	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.2-5 for RM.
8. Re-transmission of the current data message will complete or time-out prior to transmission of the next message received from the sending application.	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.2-8 for RM.
9. At a minimum, failed attempts and repeated retries will be reported to the calling application for transfer to System Messages.	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.2-3 and 1.2.2.1.2.1-22 for RM.
10. The number of data message transmission retries will be statically configurable per data stream type. (Note: Entries will be procedurally controlled to prevent exceeding the system data transmission time limitations)	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.2-7 and 1.2.2.1.2.3-4 for RM.
11. Message delivery attempts by the sender will be aborted based on a configurable expiration timer set for acknowledgment responses.	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.2-6 and 1.2.2.1.2.3-3 for RM.
12. The API will provide automatic sequential numbering of data packets based on stream or port connections	AM/CLM, RM	Retained for AM/CLM. Rewritten as 1.2.2.1.2.1-16 for RM.
13. The API will contain the capability to receive and buffer the next application message(s) until the current message is transmitted, re-transmitted or aborted.	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.3-5 for RM.

14. The receiver will discard the whole message if an error is detected or message re-assembly is unsuccessful.	RM	Not applicable to AM/CLM. Moved to 1.2.2.1.2.1-19 for RM.
15. The receiver will report receive errors to the calling application for transfer to System Messages.	AM/CLM, RM	Retained for AM/CLM. Rewritten as 1.2.2.1.2.1-20 and 1.2.2.1.2.1-22 for RM.
16. In cases where the source of the stream has moved from one platform to another or has closed a stream and then re-opened the same stream, the API will allow receiving applications to continue to receive a data stream without having to register again.	RM	Not applicable to AM/CLM. Intent of requirement is now covered for RM by 1.2.2.1.2.1-5.
17. Reliable message for connection oriented (point-to-point) communication will be provided as specified by the TCP layer standards.	AM/CLM	Retained for AM/CLM. Not applicable to RM.
18. The sender will transmit the multicast and broadcast data even though a client has not yet registered to participate in reliable messaging communication.	RM	Not applicable to AM/CLM. Rewritten as 1.2.2.1.2.3-2 for RM.
19. The API will associate the stream name (defined in the Naming Standard) with a unique multicast address. The stream name assignment to multicast addresses will consist of a static file for Redstone.	AM/CLM	Retained for AM/CLM unreliable multicast option. Not applicable to new RM design; requirement was implementation-specific.
20. The API will allow applications to open multiple streams within a single process.	AM/CLM, RM	Retained for AM/CLM. Covered for RM by 1.2.2.1.2.1-3 and 1.2.2.1.2.1-5.
21. The API will provide a function which relieves applications from having to poll for received data.	AM/CLM, RM	Retained for AM/CLM. Copied to 1.2.2.1.2.1-9 for RM.
22. The API software will be structured and designed such that a single software baseline can be obtained regardless of platform type (i.e. VxWorks or UNIX).	AM/CLM	Retained for AM/CLM. Will be retained as a design goal for RM.
23. The multicast acknowledgments will be transferred to SDC for recording.	RM	Not applicable to RM. Moved to 1.2.2.1.2.1-25 for RM.
24. The API will contain the ability to enable/disable the transfer of acknowledgments to SDC for recording, with enable as the default setting.	RM	Not applicable to RM. Moved to 1.2.2.1.2.1-26 for RM.
25. The API will provide the capability to support multiple senders on a single stream.	RM	Not applicable to RM. Covered for RM by 1.2.2.1.2.1-5.

APPENDIX D

Applicability of Requirements to Gateway Platforms

APPENDIX D: APPLICABILITY OF REQUIREMENTS TO GATEWAY PLATFORMS

The following table lists Network API requirements from section 1.2.2.1 of this document that are not applicable to CLCS Gateway platforms. All other Network API requirements are applicable to all RTPS platforms.

Requirement	**Rationale**
~~(1.2.2.1.2.1-7) The API shall allow a client process to specify a priority value for each message sent, with a range of at least two different priority levels.~~	~~CCP and CCWS applications need to send priority commands, but there are no requirements for a Gateway to send messages with an elevated priority. Responses are processed in the order that commands are received, so a GW will at least handle first the responses to any high priority commands. (The GW implementation can be much simpler without the priority requirement.)~~ *Note: This requirement is being reallocated to Gateway platforms to support sending of System Event Codes at an elevated priority. Thus, this entry will be deleted from Appendix D in future versions of this document.*
(1.2.2.1.2.1-24) The API shall allow a single host to participate in more than one activity.	Per Atlas Design Panel 1, this requirement was added for the DDP, CCP, and CCWS platforms only. There is no anticipated need for a gateway to simultaneously support more than one test set at a time.

APPENDIX E

Derivation of Maximum RM Message Size Requirement

APPENDIX E: DERIVATION OF MAXIMUM RM MESSAGE SIZE REQUIREMENT

Network Services CSC requirement 1.2.2.1.2.1-18 states that RM must handle messages of at least 50 kilobytes. This appendix describes the derivation of this requirement.

There are three SLS requirements that imply a worst-case message size:

(2.2.2.1.1) The system shall support 25,000 End-Item Function Designator changes per second continuously. This is the “system maximum data bandwidth”.

(2.2.2.1.2) The system shall support a peak of 50,000 End-Item Function Designator changes in a given second without losing any data.

(2.2.2.1.3) The system shall support 1,000 End-Item Function Designator changes during a 10 millisecond period.

The first two of these requirements both specify a rate of change data that is assumed to be evenly spread over any given second—that is, either between 100 SSR intervals or 10 DSR intervals. Of these two, 2.2.2.1.2 implies the greater message size, so we will use it for our calculations.

The third requirement is limited to a single 10 millisecond period, equal to one SSR interval. This requirement is therefore also relevant to our message size calculations. (It is meaningful for RTCN messages only.)

From the latter two of these requirements, then, we can determine a worst-case message size as follows:

$$\frac{50{,}000\text{ trans.}}{1\text{ sec}} \times \frac{8\text{ bytes}}{1\text{ trans.}} \times \frac{1\text{ sec}}{100\text{ SSR ticks}} = 4{,}000\text{ bytes per SSR tick (10 ms)}$$

$$\frac{1{,}000\text{ trans.}}{10\text{ ms}} \times \frac{8\text{ bytes}}{1\text{ trans.}} \times \frac{10\text{ ms}}{1\text{ SSR tick}} = 8{,}000\text{ bytes per SSR tick (10 ms)}$$

$$\frac{50{,}000\text{ trans.}}{1\text{ sec}} \times \frac{8\text{ bytes}}{1\text{ trans.}} \times \frac{1\text{ sec}}{10\text{ DSR ticks}} = 40{,}000\text{ bytes per DSR tick (100 ms)}$$

Note: An average transaction size of eight bytes is assumed.

From this, we see that the SLS effectively requires a worst-case RTCN message size of **8,000 bytes**, and a worst-case DCN message size of **40,000 bytes**.

Because RM will be implemented as a generic service that is adaptable to both the RTCN and DCN, the DCN figure above will be used in setting the maximum message size that RM must support. Adding a 25% safety margin gives us a general requirement for RM to support messages of up to **50,000 bytes**, as specified in CSC requirement 1.2.2.1.2.1-18.

This figure is within the range of supported UDP datagram sizes for all RTPS platforms. Therefore, no fragmentation and re-assembly of messages by RM will be necessary. CSC requirement 1.2.2.1.2.1-18 will be met solely through the use of COTS operating system and protocol features.